基于 Spark 的混合协同过滤算法改进与实现 *

王源龙[1]，孙卫真[1]，向　勇[2]

(1. 首都师范大学 信息工程学院 计算机科学与技术系，北京 100048; 2. 清华大学 计算机科学与技术系，北京 100084)

摘　要：针对传统协同过滤在推荐过程中存在的稀疏性、扩展性以及个性化问题，通过引入算法集成的思想，旨在优化和改进一种新型的基于 Spark 平台下的混合协同过滤。借鉴了 Stacking 集成学习思想，将多个弱推荐器线性加权组合，形成综合性强的推荐器。首先，算法基于近邻协同过滤，结合分类、流行度、好评度等对近邻相似度计算策略进行优化，旨在改善相似度的合理性以及相似度计算的复杂度，在一定程度上改善了评分稀疏性的问题；同时，该算法结合 Spark 分布式计算平台，充分借鉴分布式平台的优点，利用其流式处理以及分布式存储结构等特性，设计并实现了一种推荐算法的增量迭型，解决了协同过滤算法扩展性和实时性问题。实验数据采用 UCI 公用数据集 MovieLens 和 NetFlix 电影评分数据，实验结果表明，改进算法在推荐个性化、准确率以及扩展性上都有不错的表现，较以前同类型算法均有不同程度的提高，为推荐系统的应用提供了一种可行的算法集成方案。

关键词：集成学习；协同过滤；稀疏性；扩展性；Spark 流式计算；增量模型；分类

中图分类号：TP301.6　　doi: 10.3969/j.issn.1001-3695.2017.10.0933

New improvement and implementation of hybrid collaborative filtering algorithm based on Spark platform

Wang Yuanlong[1], Sun Weizhen[1], Xiang Yong[2]

(1. *College of Information Engineering*, *Capital Normal University*, *Beijing* 100048, *China*; 2. *Dept of Computer Science & Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

Abstract: Aiming at optimizing and improving a hybrid collaborative filtering based on spark platform for its sparsity, scalability and personalized recommendation by using the method of algorithms integrated. This paper takes the model of Stacking algorithm integrated to integrate multiple weak recommender units in a linearly weighted into a comprehensive recommender. Firstly, this article optimizes the collaborative filtering based on the nearest neighbor by presorting and adjusting the similarity calculation strategy with popularity and praise degree, and improves the rationality and complexity of similarity calculation. It solves the problem of score sparsity to some extent. At the same time, this algorithm integrates closely distributed computing platform, which can make full use of the advantages of distributed platform to design and implement an increment iterative model of recommendation algorithm by using the Spark streaming and distributed storage structure and it solves the problem that collaborative filtering algorithm is hard to expand and make poor real-time performance. The experimental data uses UCI public data set named MovieLens and NetFlix films' score, and the experimental results show that the improved algorithm has a good performance and makes great progress in personalized recommendation, accuracy and scalability compared with the previous algorithms. It provides a feasible algorithm integration scheme for the application of the recommended system.

Key Words: integrated learning, collaborative filtering; sparsity; extensibility; spark streaming; incremental model; classification.

0　引言

在互联网与物联网相交的大数据时代的背景下，信息过载成为了目前互联网用户所面临严峻的问题，面对海量的数据，本文如何准确高效地寻找到需要的信息已经变得越来越难，而推荐系统是解决这一问题的有效手段。

推荐系统[1]，是提供信息推荐的系统，以降低用户寻找有效信息的难度，增加用户与网络信息交互的体验感为目的的算法本

基金项目：北京市教委科技计划项目（KM201310028014）

作者简介：王源龙（1988-），男，硕士研究生，主要研究方向为数据挖掘和大数据处理方向；孙卫真（1963-），男，副研究员，主要研究方向为操作系统、图像处理、数据挖掘；向勇，副教授，博士，主要研究方向为计算机协同工作、操作系统、计算机网络、数据挖掘.

体。推荐系统的核心是推荐算法，推荐算法与其他领域认知科学、信息检索、数学、社会科学等，有深厚的渊源，是一个多学科交叉的研究方向。

推荐系统始于上世纪九十年代，历经了二十多年的发展后，其理论也在不断地更新和完善，应用也变得越来越广泛，尤其是在当今互联网与物联网相交汇的时代，许多大型的电子商务平台，搜索引擎，新闻门户网站，社区交友平台，像亚马逊、京东、雅虎、微博、Netflix 等大型商务、娱乐网站，无一不存在推荐系统的影子。此外，其他领域诸如智能交通、智能家居、人工智能等，推荐系统也开始越来越多地涉及和应用。所以，当下对推荐系统的研究就变得迫切而有意义[1]。

1 相关工作

传统的推荐方法通常分为两大类：基于协同过滤(CF)[1,2]和基于内容方法(Content-based)[1,3]，其中，基于协同过滤的方法为本文的研究重点。

基于内容的推荐[3]，是一种利用用户历史数据记录的细粒度的推荐，推荐信息的载体往往是基于内容本身，如新闻，Web 文档等，优点是推荐内容较为准确，缺点是推荐受内容的局限，推荐内容无法做到新颖，也不利于保护隐私。

基于协同过滤的算法有基于近邻[4,5]和基于模型[6,7]两类算法。协同过滤的基本思想是：集体智慧学习原则，通过朋友的行为来间接判断你的行为。协同过滤最大的优点是能够挖掘出用户潜在可能感兴趣的物品，使推荐内容具有新颖性，同时也保护了用户的隐私。

基于近邻的协同过滤具体可基于用户和基于项目两种，基于用户协同过滤[4]是通过分析用户评分来推测用户喜好，基于物品的协同过滤[8,9]是通过分析物品的相似性来做推荐，而基于物品的推荐是利用物间相似度大小为用户进行推荐。基于模型的方法尝试更进一步地填评分矩阵，使用一些机器学习算法来对评分向量进行训练，然后建立模型来预测用户对于新的物品的得分。协同过滤算法面临的两个最大问题是如何解决评分稀疏性以及算法扩展性问题。

本文的研究内容将从分布式计算的角度，围绕着两个问题进行展开。此前，研究人员提出过一些协同过滤改进算法。如基于用户协同过滤算法的推荐效率和个性化[10]，做到了比较好的个性化需求，但过于强调效率和个性化，没有解决用户推荐算法的最大的问题--扩展性；基于矩阵分解的协同过滤，典型的算法有 SVD[11]，这类算法是解决稀疏矩阵的一种有效的手段，它的最大优点是快，但它同样没有解决算法扩展性的问题；基于内容的推荐算法在算法扩展性上做得很好，但往往容易让推荐缺乏新颖性，降低推荐的意义和价值；基于 Spark 平台的协同过滤推荐算法[12]，借助于分布式平台的海量数据计算能力，但在算法上并没有过多改进；现在关于推荐算法 Spark 下唯一实现的是基于最小二乘的推荐算法，但其在扩展性以及实时性上表现不足。

所以，所以本文寻求的目标是在个性化，新颖性以及扩展性之间达到一种期望的最大化，力求能够在一两个方面能够较以前的算法有所突破。本算法有三个创新点：首先本文提出算法实现平台是基于 Spark 的分布式平台，这与单机的模型实现细节有所不同，借助于 Spark 集群，使得算法的并行能力大大提高；其次，本算法成功将基于用户，项目和基于模型的推荐算法优点有效地切合在一起，可以实现更好的推荐个性化效果。为了降低算法实现难度，本算法主要针对的是协同过滤算法进行改进，只对用户的行为以及评分数据进行预测，不对基于内容的推荐相关涉及。第三，本算法实现了基于增量模型算法，可以迭代计算做实时推荐，这是本算法的一大亮点。本算法由于是多种算法思想的融合，为降低本文算法实现的难度，决定不加入基于内容的推荐算法。综上所述，本算法是考虑了基于用户、项目和模型三者的结合，利用预先分类降低相似度计算量，利用流行度、好评度指标对用户相似度与物品相似度进行权值调整，得到基于用户的推荐列表和基于物品的推荐列表，最终于最小二乘法推荐进行加权融合，这种模型和一种随机森林模型有点类似，本文也正是基于这种弱分类器线性组合构成强分类器的思想，最终求出的模型是一种增强的混合推荐模型。

2 改进算法详细设计

2.1 算法改进思路

本文主要运用算法集成的思想，对多个协同过滤进行合理优化与结合，取长补短，以期达到最大化算法的运行效率和预测的准确性。概况而言，本算法在实现的过程中主要做了如下的工作：对数据对象进行预分类、简化无用相似度计算、提高相似度的合理性、利用流式计算来实现增量模型、算法与平台的集成等。

在平台层面，采用 Spark 分布式计算平台，利用分布式高可靠、高性能的特点，除了给算法的运行提供强大的计算能力支持，最大的作用是利用 Spark 流式处理模型完成了算法增量模型的计算，改善了协同过滤算法存在的扩展性问题。

在算法方面。首先用预先分类技术解决评分稀疏性带来的大量无用相似度计算和存储难题。具体描述为：在计算相似度之前先将物品以及用户按照标签进行分类，保存用以对增量数据进行快速的类别判断，提供快速的模式匹配，完成模糊推荐。接着对于已经分组的用户，在组内采取建立倒查表等方式，进一步缩小两两相似度的计算次数，并进行原始相似度的计算。原始相似度计算完成，利用流行度以及好评度对上一步计算的用户相似度进行调整，规则是：降低流行度高的物品对应的用户间的相似度，提高好评度高的物品对应用户间的相似度。得到用户相似度后，跟据用户相似度确定合适的近邻，按照其与近邻相似度以及其评分，对每个用户计算各自的推荐物品评分，形成推荐列表。接着运用基于项目的协同过滤，得到一个推荐列表，同理运用矩阵分解的方法也计算出一个推荐列表，最后做线性加权平均得到最终的推荐列表。这样做的理由是，若三个推荐列表中都有的项，则最终计算出的评分一定会比较高，若只有其中一个推荐，

则说明推荐的可信度不高，评分显然会降低，这样不仅可以为推荐提供很好的可解释性，还能提高为用户推荐的质量。模型建立以后，通过交叉验证对模型进行评估与选择。

2.2 改进推荐算法运行流程

基于 Spark 平台的推荐算法流式计算过程，可以描述为：Spark 分布式程序启动后，首先通过上下文加载驱动，获取并分配计算资源，启动守护进程；然后 Spark Streaming 从 Kafka 消息队列中读入实时数据，进行实时数据归类，分别对用户相似度，物品相似度进行实时计算，得到推荐列表；将基于最小二乘的矩阵分解得到的推荐列表与之前得到的推荐列表加权合并，得到并更新最终推荐列表以及其预测评分。如图 1 所示。

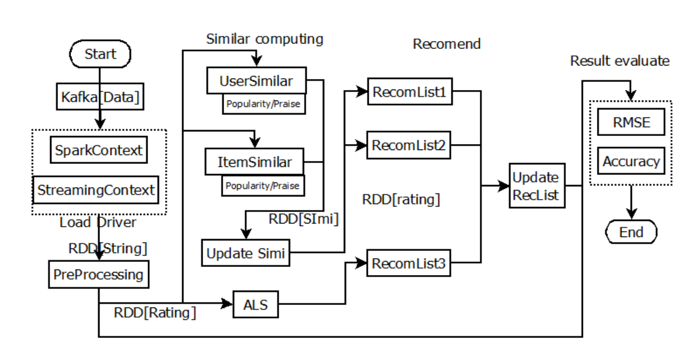


图 1 Spark 推荐系统运行流程

2.3 算法详细设计

2.3.1 平台从单机转向集群

单机转向集群的基本思路是：利用开源分布式框架将廉价 PC 机逻辑地连接起来，构成一个逻辑整体，主从式的控制结构，主节点对从节点进行任务调度、分发和容错，从节点实现并行计算，此结构被证明是一个拥有高可靠、高并发，高性能计算能力的分布式结构。实验集群环境由五台物理机组成的集群实战环境，集群物理结构图 2-a，集群逻辑结构图如图 2-b。

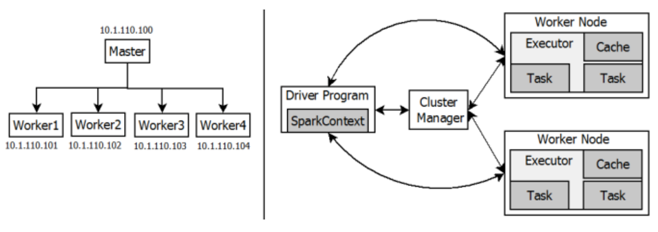


图 2 a 集群物理结构图 b 集群逻辑结

2.3.2 从离线推荐变为在线推荐

传统推荐往往是离线算法较多，但本文却同时实现了在线与离线计算，借助了第二代分布式数据处理架构 Spark，以 Kafka[13]作为消息队列提供源数据，运用 Spark Streaming 流式处理实现增量计算模型；离线部分主要使用 RDD，Spark SQL，Spark MLlib[14]等技术进行了高效的矩阵运算和机器建模；以 HDFS 作为分布式文件系统持久化基础、以 HBase[15]作为存储数据库；HDFS 提供了集群本地数据持久化的载体[16]。图 3 展示了实时与离线计算的流程结构图。

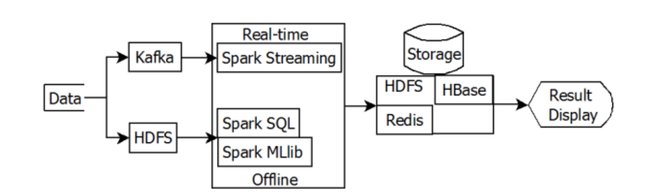


图 3 实时与离线结构图

2.3.3 用户、项目预分类

对于协同过滤算法，即对用户行为进行预测的一种粗粒度推荐算法。其特征数据主要有用户评分，用户属性，项目属性，用户点击率，停留时间等等。本文以 MoviesLen 和 NetFlix 电影评分数据作为主要的输入特征数据源。评分数据最为典型的就是，其的格式为“用户 id:电影 id:评分:时间戳”，这种数据的缺点是数据信息量大量被隐藏，需要深度挖掘，而且数据被集中在一起，不易看出他们之间的关系。所以，在计算相似度前，先考虑借助用户属性与项目属性对不同的用户以及物品进行简单分类。这样一个举措带来的作用非常的多，首先降低相似度计算的复杂度；其次是降低大评分数据矩阵的稀疏性问题；同时可以为粗粒度的实时推荐带来便利，这一点尤为重要，它解决了协同过滤的冷启动问题。

对用户进行分类过程如下：将“用户-物品-评分”表建立倒查表以对用户进行分类，如图 4 所示，基本分类思想是将具有相同评分项的用户或者项目归为一类。没有评分项的用户或者物品本文称为孤立项，将其归为一类，以备后续算法推荐做准备。注意，在用户类或者项目归类的时，需要对其进行编号，并且赋予编号实质的意义，并在用户与项目的不同编号之间建立某种联系，以方便对用户进行实时快速推荐。对物品的分类过程与用户原理相同。图 5 描述的分类过程用户与项目的分类过程。

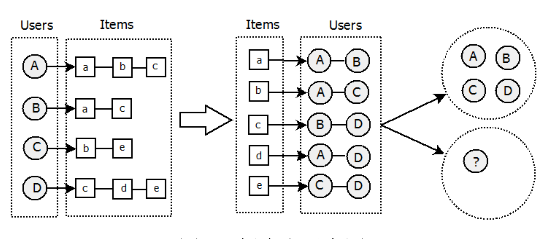


图 4 倒查表示意图

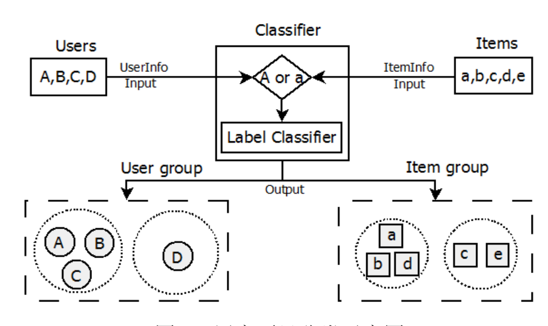


图 5 用户项目分类示意图

2.3.4 相似度计算增量模型

对于基于用户的协同过滤，除稀疏性问题，最大的瓶颈在于

在线推荐产生的庞大相似度的计算。对于每一次增量所有相似度都需要重新计算，计算开销太大，所以考虑在此需要做一些相似度计算策略的调整，目标是保证在对相似度影响最小的情况下最大限度降低重复相似度的计算量。对于物品的协同过滤亦如此。因此，本算法在相似度计算策略上做了如下调整：

a)忽略增量数据对原始对象间相似度的影响。由于增量数据相对原始数据较小，对原对象间相似度影响程度一般比较小，因此在计算相似度时不应该再重复计算原始对象间的相似度，直接读取持久化原始对象相似度是最有效的方式。而对于新增的数据，需要计算的是新增对象间的相似度以及新增对象与原始对象间的两两相似度。具体计算策略是：增量输入，判断增量 id，若对象已存在于系统，则该对象不参与本次相似度计算，将其持久化到数据库，并且将对象出现的次数加 1,当出现次数到达一定数量时，读取相关数据并参与相似度的计算。

b)相似度并行化处理。此处并行得以实现，得益于对数据对象的预分类，每一个类簇内部联系很紧密，类簇间联系是松散的，类簇间的相互制约影响已经被降到最低，形成了一种极适合分布式并行计算的数据格式，实现了所有的类簇的并行计算。

c)相似度的更新与持久化。增量相似度计算完毕后，不断地与先前已存在的相似度进行合并持久化到数据库，若有相同即更新，若无相同则添加。持久化内容包括元数据，计算后的相似度，以及一些中间状态变量(相似度更新时间、唯一标识符、推荐列表更新时间以及对应唯一标识符)等。图 6 描述了相似度与推荐模型的更新过程。

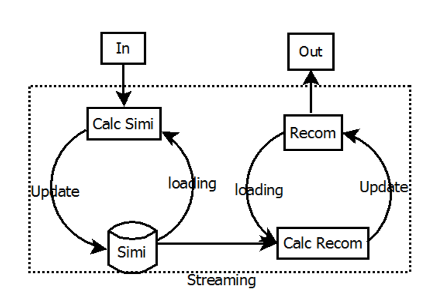


图 6　模型更新示意图

d)相似度计算次数简化分析。设原数据库中已存在用户数为 m，增量用户数为 n，则进行两两全组合相似度计算次数为 $S1_{count}$，如式(1)；简化后相似度计算次数后相似度计算的次数为 $S2_{count}$，如式(2)：

$$S1_{count} = C_{m+n}^{n} = \frac{(m+n)\times(m+n-1)\ldots\times m}{\underbrace{n\times(n-1)\times(n-2)\ldots\times 1}_{n项}} \tag{1}$$

$$S2_{count} = n\times m + C_n^2 = n\times m + \underbrace{n\times(n-1)/2}_{2项} \tag{2}$$

从式(1)(2)可以推测，当原用户 m>>n 的时，公式时间复杂度分别是 o(m^2)和 o(m),故而 $S2_{count} << S1_{count}$，也就是说，即使当 m 很大的时候，改进后的相似度的计算次数数量级将远远小于简化前。虽然方法确实在当前牺牲了少量相似度精度，但是却满足了增量计算模型对数据量的需求，这是一种精度与时间复杂度的合理平衡。

2.3.5 相似度计算方法

对于基于近邻算法的协同过滤，往往是通过计算对象间相似度，以相似物品或相似用户的高分物品来进行推荐。相似度的计算方法比较多，不同的相似度测量方法对推荐的结果影响比较大，针对用户相似度而言，目前认为皮尔逊相似度比较好，因为它相比余弦相似度多了归一化操作，这样减小了不同用户打分的习惯所带来的相似度的偏差，对于物品间的相似度，一般经验是采用余弦相似度。本文中借鉴了这两种相似度计算方法的原型，并且在这两种计算计算方法的基础上，引入了流行因子和好评度等因子分别对某些相似度进行权值调整，权值调整的目的是为了更加准确地度量相似程度。各相似度的计算公式如下：

$$Cos(x,y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} = \frac{\langle x,y\rangle}{\|x\|\times\|y\|} \tag{3}$$

$$Pearson(x,y) = \frac{\sum_i (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2}\sqrt{\sum(y_i-\bar{y})^2}} = \frac{\langle x-\bar{x}, y-\bar{y}\rangle}{\|x-\bar{x}\|\times\|y-\bar{y}\|} \tag{4}$$

$$Simi_{users}(x,y) = Pearson(x,y) + \alpha\times P + \beta\times Q,\ (x,y\in Set(users)) \tag{5}$$

$$Simi_{items}(x,y) = \cos(x,y) + \alpha\times P + \beta\times Q, (x,y\in Set(items)) \tag{6}$$

式(3)为原始余弦相似度，为正值，用于计算物品相似度；式(4)为皮尔逊相关性,用于计算用户相似度，其中 x,y 表示用户 id；式(5)(6)分别表示调整后的用户相似度和项目相似度，其中 P,Q 分别表示流行度和好评度，为(-1,1)，正表示相似度正向补偿，负表示相似度惩罚；α,β 分别为流行度阈值系数和好评度参数，正值。

2.3.6 评分的预测

由于基于用户的协同过滤推荐能够很好的发现用户潜在的兴趣，而基于物品的协同过滤推荐能够对根据用户行为实时更新推荐结果，基于最小二乘法的矩阵分解法能够较好处理评分矩阵稀疏性等问题，并且在评分的准确度上最小二乘一直是一个 Baseline，所以本文在完成了前两者结合的同时，也将最小二乘进行了融合，为的是充分发挥各自算法的优势。最终对物品的评分结果由基于用户相似度的预测评分、基于物品相似度的预测评分以及基于最小二乘法的矩阵分解评分三部分线性加权求得。

那么对于没一个单独的推荐算法的预测评分的总体计算规则，本文采用的相似度乘以相似对象（用户或物品）的评分再除以相似度的总和。

设用户 u 对物品 i 的预测评分 v 则预测评分的求解算法描述如下：

计算基于用户的协同过滤推荐列表

a)找到用户 u 的 k 个近邻。

b)找出每一个 u 的 k 个近邻用户对项目 i 的所有真实评分组

成的集合，记录为 Rating(x,i,v)，其中 x 表示对 i 有过评分的 u 的 k 个近邻。

c. 利用相似度与真实评分相乘求和除以总相似度，则用户 u 对项目 i 的预测评分如式(7)：

$$\text{Predict}(u,i,\bar{v})_{user}=\frac{\sum[Simi_{users}(x,u)\times Rating(x,i,v)]}{\sum Simi_{users}(x,u)},\quad x\in Set(users) \tag{7}$$

计算基于项目的协同过滤推荐列表

a)找到用户 u 评分最高的 k 个项目

b)找出与这 k 个项目最相似的 n 项目

c)利用相似度与真实评分相乘求和除以总相似度，得出用户 u 对物品 i 的评分。如式(8)：

$$\text{Predict}(u,i,\bar{v})_{item}=\frac{\sum[Simi_{item}(x,u)\times Rating(x,i,v)]}{\sum Simii_{tem}(x,u)},\quad x\in Set(items) \tag{8}$$

d)基于最小二乘法的矩阵分解，通过损失函数最小化，迭代求解最优隐含特征矩阵 U、V 来得到用户 u 对物品 i 的评分,求出最优的矩阵。式(9)(10)分别为最小二乘法的损失函数和预测评分表达式。

$$\min\left\{f(U,V)=\left(\sum r_{i,j}-u^{T}v\right)^{2}+\lambda\left(\sum n_{ui}u_{i}^{2}+\sum n_{mj}v_{j}^{2}\right)\right\} \tag{9}$$

$$Predict(u,i,\bar{v})_{als}=U_{u}\times V_{i},u\in Set(users),i\in Set(items) \tag{10}$$

e)最终的推荐结果 Predict($u,i,\bar{v}$)的预测评分等于基于用户、基于项目以及最小二乘法计算三者的平均预测评分。评分表达式如式(11)所示，其中 ω,φ,η 分别是其推荐的权重。

$$\begin{aligned}&\text{Predict}(u,i,\bar{v})=\omega\times\text{Predict}(u,i,v)_{user}+\varphi\times\text{Predict}(u,i,v)_{item}\\&+\eta\times Predict(u,i,\bar{v})_{als},\quad \omega+\varphi+\eta=1,u,x\in Set(users),\\&i\in Set(items),v\in Set(pref)\end{aligned} \tag{11}$$

f)评分的更新。与相似度更新一样，在进行推荐更新之前，需要对推荐评分列表持久化。评分列表更新的原则是，更新前后两次时间间隔内相似度有变化后的那些对象类集，重新计算其推荐评分列表、更新并进行持久化。对于相似度没有变化的这部分推荐列表，直接从数据中读取相关的记录进行推荐。

3 实验以及其结果分析

3.1 测试数据集

实验数据采用了 UCI 公开 MovieLens 的数据集以及 NetFlix 数据集，分别对 100K(10 万条) 、1M(约 100 万条)、10M(约 1000 万条数据）和 100M(约 1 亿条数据)进行了四组实验。评分范围是（1-5），数值越大表示评价越高，其中主要用到的评分数据表。文件数据格式表 1 所示。

表 1 数据格式

UserID	MovieID	Rating	Timestamp
用户编号	电影编号	评分	时间戳

3.2 参数设置与调试

本实验所有关于算法的参数的调试与配置，均只对 MoiveLens 数据集以及 NetFlix 电影评分数据集负责。其他数据集不保证能得到类似结果。

表 2 算法参数说明

基于近邻推荐参数					最小二乘推荐参数		
用户分类数量	物品分类数量	流行度阈值	好评度阈值	近邻阈值	隐藏因子个数	迭代次数	正则项系数
8	16	>1/5	>4.5	>0.6	20	50	0.1

3.3 结果评估

1）均方差根误差

对于评分的预测准确度，常采用均方根误差(RMSE)来评价。误差越小，预测值与实际值相差越小，准确度越高。其公式为

$$RMSE=\sqrt{\frac{\sum_{u,i\in T}(r_{ui}-\hat{r}_{ui})^{2}}{|T|}} \tag{12}$$

其中：r_{ui} 表示用户 u 对电影 i 的实际评分，r_{ui} 是通过推荐算法预测的评分，|T|为测试集总量。

2） 准确率与召回率以及 F1

一般用准确率和召回率来评估推荐结果的好坏程度。准确率表示推荐结果中用户真正感兴趣物品占推荐物品总数的比例，召回率一般是指推荐物品中用户感兴趣的占用户所有感兴趣的物品的比例。准确率与召回率两者一般呈现负相关的趋势，因此，两者的调和平均数来F1可以更加全面地衡量整体推荐结果的好坏。下面分别是准确率，召回率以及调和平均值的计算公式。

$$P_{mean}=\frac{\sum_{i\in U}\frac{A_i}{A_i+B_i}}{|U|},\quad R_{mean}=\frac{\sum_{i\in U}\frac{A_i}{A_i+C_i}}{|U|},\quad F1=\frac{2P_{mean}\times R_{mean}}{P_{mean}+R_{mean}} \tag{13}$$

其中：A_i 为某用户推荐并且是用户感兴趣的物品，B_i 表示为某用户推荐但并不感兴趣的物品，Ci 表示某用户感兴趣但没推荐的物品，|U|为参与评估的项目数量，P_{mean} 代表平均准确率，R_{mean} 代表平均召回率，F1 是准确率与召回率的调和平均值，F1 值越大，表示准确度和召回率都越高。

3.4 实验结果与分析

在本文中，暂且将本文改进算法命名为 HybirdCF，其相关实验与测试结果如下。本实验将从几个维度将对本算法进行评估。

a）各算法的均方根误差。各个算法的参数均为本实验环境下的最优参数的情况下，改进的混合协同过滤算法与几种经典协同过滤算法的预测评分的均方差误差如图 7 所示。

从图 7 中可以看出，在同等情况下，改进的算法比单纯基于用户的协同过滤和单纯基于物品的协同过滤均方根误差更加小，也比 Spark 下自带的最小二乘法的均方根预测误差小，预测

更加准确，说明算法在预测准确度上有提高。

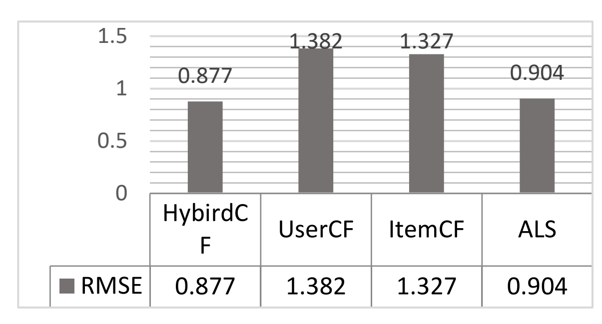


图 7　HyBirdCF 与传统算法的均方根误差

b)不同数据规模的均方根误差。算法参数包括分类数量、用户相似度阈值、物品相似度阈值、流行度阈值、好评度、近邻阈值、误差推荐阈值。图 8 为算法在各类参数最优条件下，不同数据规模的均方根误差的实测情况。

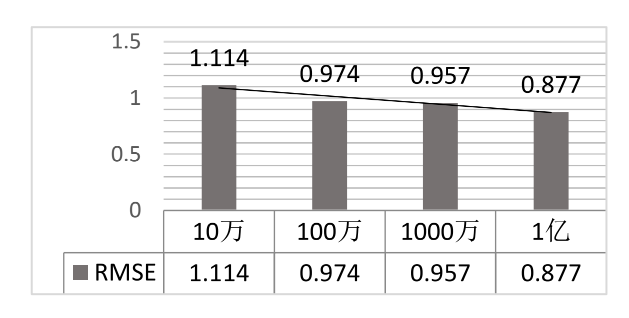


图 8 不同数据规模对 HyBirdCF 评分误差的影响

从图 8 中可以看出，改进算法在数据量越大的时候，评分误差会越小。这意味着，评分数据越多，本算法将可能会得到更加准确的结果，所以也要防止过度拟合的情况。

c)改进算法在不同增量规模数据下实时性。如图 9 所示，横坐标表示数据实时输入的数据流数据规模，纵坐标表示时间，该图反映了数据规模对流处理增量模型的计算耗时影响。

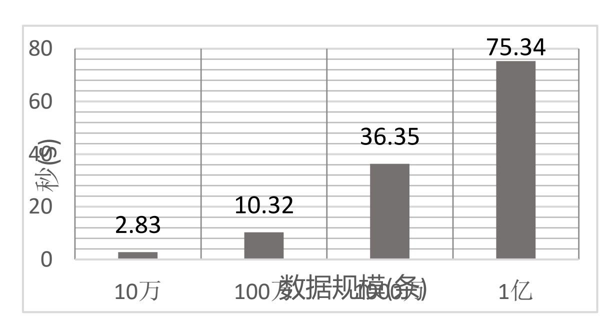


图 9　增量数据的耗时影响

图 9 反映的是增量数据对系统推荐的实时性处理情况。图中的数据是每一秒流入推荐系统的评分条目数。可以看出，对于配置为 5 台酷睿 4 核心主频 3.5G，8G 内存的 PC 机，效果比较让人满意。也依托于强大的处理能力和实时计算的处理机制，才使得协同算法扩展性问题得到了很大的改善。在实际情况中，随着集群规模的扩大，实时性会得到更进一步的提高。

d）在算法模型参数固定不变的情况下，改进算法在不同数据规模下推荐结果准确率、召回率的变化曲线图如图 10 所示。

	1万	10万	100万	1000万	1亿
Precise	0.003	0.029	0.283	0.544	0.747
Recall	1.000	0.931	0.862	0.684	0.580
F1	0.003	0.027	0.244	0.372	0.433

图 10　准确率、召回率以及 F 值变化趋势图

图 10 中，三角表示准确率，方形表示召回率，圆形表示的是准确率和召回率的调和平均数，可以看出，推荐准确率在随着评分数据量的不断增大而增大，而召回率却不断在减小，F1 值在缓慢地上升。这说明在模型参数不变的条件下，准确率与召回率在一定程度上呈现负相关的趋势，这与前文的推测是相符的，即准确度会随着评分数量的增多而提高，但相反召回率会相应降低，表明随着用户感兴趣的物品的增多，推荐的结果将会更加准确，推荐内容的新颖性将会逐渐地降低，但 F1 会随着数据的增加而不断增大。

4　结束语

本文算法从算法层面与平台层面，均体现出了集成算法学习的思想，总结起来，可概括为以下两方面：

a)从算法的设计上层面。为降低相似度计算的复杂时间复杂度，减少评分的稀疏性，增加分布式并行计算的可能可能性。在相似度的计算上，运用物品流行度、好评率等因子对用户相似度以及物品相似度分别进行调整和优化，使其与实际的相似性更加接近。在预测评分上，利用近邻的 TopN 的相似度以及评分加权求和分别求出每个用户的预估评分推荐列表。将上述两推荐列表与基于最小二乘的推荐列表三个推荐列表加权求和，得到最终的推荐列表。其中三个权值分别代表不同方向维度上的权重，可实现动态参数调节，若需要突出某一个维度特性，可进行实时参数调节，得到最优的推荐参数，若想发现用户的某些偏好，亦可以调节这三个维度上的权值。

b)从算法运行的平台上。基于 Spark 分布式计算平台，利用 Spark Streaming 流式处理机制，实现多种协同过滤混合的增量推荐模型，包括相似度增量模型和推荐增量模型。其中，相似度增量模型是不断地将新的相似度与旧相似度归并持久化，增量的计算对持久化量不影响；推荐模型亦同，不同的是推荐内容的更新是依赖于相似度的更新，对每个用户的推荐也一样。

本实验所用数据为公用数据集 MoivesLens 和 NetFlix 电影评分数据。从横向来看，算法优化与设计相对合理，从纵向来看，算法在给定测试集上的效果比预期的要好。实验结果表明，改进算法相比于已往此类型的推荐算法，有更好的个性化效果，更小的预测误差，更实时的推荐效果，更灵活的扩展性，在评分预测和准确度以及扩展性上都有不同程度的改善和提高。

本算法为基于算法集成的混合型协同过滤，特别适用于以用户画像为基础推荐场景协同过滤推荐，诸如商品推荐、音乐推荐、美食推荐，旅游推荐等诸多领域。

参考文献：

[1] Francesco ricci, LiorRokach, Bracha Shapira, etc. Recommender systems handbook [M]. New York: Springer, 2011, 1 (1): 39-184

[2] Cheung K W, Tian L F. Learning user similarity and ratings for collaborative recommendation [J]. Information Retrieval, 2004, 7 (3-4): 395-410

[3] Balabanovic, M. Shoham. Content-based Collaborative Recommen dation [J]. Communications of the Association for Computing Machinery, 1997, 40 (3): 66-72

[4] 王成，朱志刚，张玉侠，等. 基于用户的协同过滤算法的推荐效率和个性化改进 [J]. 小型微型计算机系统, 2016, 37 (3): 428-432

[5] 谭云志，张敏，刘奕群，等. 基于用户评分和评论信息的协同推荐框架 [J]. 模式识别与人工能, 2016, 29 (4): 359-366

[6] Zhang Yu Cheng Jiujun. Study on recommendation algorithm with matrix factorization method based on MapReduce [J]. Computer Science. 2013, 1 (1): 19-23

[7] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model [J]. Procedia Computer Science ACM, 2008, : 426-434

[8] Deshpande. M, Karypis, G. Item-based top-N recommendation algorithms [J]. ACM Trans on Information Systems, 2004, 22 (1): 143-177

[9] Linden, G. Simth, B, York. Amazon recommendations: Item-to-item collaborative filteing [J]. IEEE Internet Computing, 2003, 7 (1): 76-80

[10] 吴毅涛，张兴明，王兴茂，等. 基于用户模糊相似度的协同过滤算法 [J]. 通信学报, 2016, 37 (1): 198-206

[11] 徐新瑞，孟彩霞，周雯，等. 一种基于局部结构的改进奇异值分解推荐算法. 电子与信息学报. 2013. 635 (6): 1284-1288

[12] 胡俊，胡贤德，程家兴. 基于 Spark 的大数据混合计算模型 计算机系统应用 2015, 24 (4)

[13] 关于 kafka 消息管理系统的原理以及应用 [DB/OL]. http://kafka. apache. org/documentation/#introductio.

[14] Spark 流处理机制以及简述 [DB/OL]. http://spark. apache. org/docs/latest/streaming-programming-guide. html

[15] HBase 列式分布式数据库存储原理机制与应用 [DB/OL]. https://hbase. apache. org/book. html

[16] 陈吉荣，乐嘉锦. 基于 Hadoop 生态系统的大数据解决方案综

[17] 述. 计算机工程与科学, 2015 (10) 35.